APPENDIX

The following code listing shows one implementation of the conventional VSCALE routine in accordance with ITU (International Telecommunication Union)-T Recommendation G.728 – Annex G.

```
;search for maximum positive and negative values in vector
                        movs.w @r4+,y0
      pcopy y0,y1.      movx.w @r4+,x1
      pcmp x1,y0
  dct pcopy x1,y0
      pcmp x1,y1
  dcf pcopy x1,y1       movx.w @r4+,x1
      pcmp x1,y0
  dct pcopy x1,y0
      pcmp x1,y1
  dcf pcopy x1,y1       movx.w @r4+,x1
      pcmp x1,y0
  dct pcopy x1,y0
      pcmp x1,y1
  dcf pcopy x1,y1       movx.w @r4,x1
      pcmp x1,y0        movx.w @r4+r8,x0
  dct pcopy x1,y0       movx.w @r4+r8,x0
      pcmp x1,y1        movx.w @r4+r8,x0
  dcf pcopy x1,y1       movx.w @r4+r8,x0

      sts   y0,r1
mov   r1,r0
      sts   y1,r7
      or    r7,r0
      tst   r0,r0
      bt    VS_ZERO

      pabs  y1,y1
      pclr  a0
      pinc  a0,a0
      lds   r6,y0
      psha  #16,y0
      psha  a0,y0,a0

      sts   y1,r0
      cmp/ge      r0,r1
      bt/s  vs_pos
      mov   #0,r0

      sts   a0, r3
      neg   r3,r3
      mov   r3,r2
      shll  r2
      cmp/ge      r2,r7
      bf    vsloop3

      cmp/gt      r7,r3
      bt    vs_end2

;Case 3: maximum negative value still has room for normalization
      .align      4
vsloop41:
      shal  r7
      cmp/gt      r7,r3
      bf/s  vsloop41
      add #1,r0
```

```
        lds    r0,y0
    psha #16,y0
        movs.w @r4+,x1
        psha  x1,y0,a0     movx.w @r4+,x1
    psha  x1,y0,a1     movx.w @r4+,x1
                           movs.w a0,@r5+
                           movx.w a1,@r5+
        psha  x1,y0,a0     movx.w @r4+,x1
    psha  x1,y0,a1     movx.w @r4+,x1
                           movx.w a0,@r5+
        psha  x1,y0,a0
    movx.w a1,@r5+
                           movx.w a0,@r5+

        rts
        nop

    ;Case 2: maximum negative value exceeds minimum range vsloop3:
        cmp/ge      r2,r7
        bt     vs_end2
        .align      4
    vsloop31:
        shar   r7
        cmp/ge      r2,r7
    bf/s   vsloop31
        add    #-1,r0

        lds    r0,y0
        psha   #16,y0
        movs.w @r4+,x1
        psha   x1,y0,a0     movx.w @r4+,x1
        psha   x1,y0,a1     movx.w @r4+,x1
                            movs.w a0,@r5+
                            movx.w a1,@r5+
        psha   x1,y0,a0     movx.w @r4+,x1
        psha   x1,y0,a1     movx.w @r4+,x1
                            movx.w a0,@r5+
        psha x1,y0,a0
                            movx.w a1,@r5+
                            movx.w a0,@r5+

        rts
        nop

    ;Case 1: zero input vector
    VS_ZER0:
        pclr a0
                            movs.w a0,@r5+
                            movx.w a0,@r5+
                            movx.w a0,@r5+
                            movx.w a0,@r5+
                            movx.w a0,@r5+
        mov r6,r0
        add #1,r0

        rts
        nop

        .align      4
    vs_pos:
```

```
        sts    a0,r2
        mov    r2,r3
add     #-1,r3
        add    r2,r3

cmp/ge        r1,r3
        bf     vsloop5

cmp/ge        r2,r1
        bt     vs_end2

;Case 5: maximum positive value still has room for normalization
        .align      4
vsloop61:
        shal   r1
        cmp/ge      r2,r1
        bf/s   vsloop61
        add #1,r0
vs_end2:
        lds    r0,y0
        psha   #16,y0
        movs.w @r4+,x1
        psha   x1,y0,a0     movx.w @r4+,x1
        psha   x1,y0,a1     movx.w @r4+,x1
                            movs.w a0,@r5+
                            movx.w a1,@r5+
        psha   x1,y0,a0     movx.w @r4+,x1
        psha   x1,y0,a1     movx.w @r4+,x1
                            movx.w a0,@r5+
        psha   x1,y0,a0
                            movx.w a1,@r5+
                            movx.w a0,@r5+

        rts
        nop

;Case 4: maximum positive value exceeds maximum range
vsloop5
        cmp/ge      r1,r3
        bt     vs_end2

        .align      4
vsloop5:
        shar   r1
        cmp/ge      r1,r3
        bf/s   vsloop51
        add    #-1,r0

        bra    vs_end2
        nop
```

The following is an algorithm in accordance with a first embodiment of the present invention.

```
;search for minimum NLS
                            movs.w @r4+,x0
        pdmsb x0,a0         movx.w @r4+,x0
        pdmsb x0,y0
        pcmp a0,y0
   dct  pcopy y0,a0         movx.w @r4+,x0
```

```
        pdmsb x0,y0
        pcmp a0,y0
      dct pcopy y0,a0         movx.w @r4+,x0
        pdmsb x0,y0
        pcmp a0,y0
      dct       pcopy y0,a0         movx.w @r4,x0
        pdmsb x0,y0         movx.w @r4+r8,x1;dummy movx to reset r4=&IN[0]
        pcmp a0,y0          movx.w @r4+r8,x1
      dct pcopy y0,a0         movx.w @r4+r8,x1
        psha #-16,a0        movx.w @r4+r8,x1

        sts   a0, r0                ;r0=NLS_MIN

    ;Case 1: zero input vector
        cmp/eq      #31, r0
        bf/s  VSCALE1_check_NLSeq31_end
        mov   r6, r7        ;r6=MLS

        mov   r6, r0
        add   #1, r0                ;set r0=NLS = MLS + 1
        pclr a0
                            movs.w a0,@r5+
                            movx.w a0,@r5+
                            movx.w a0,@r5+
                            movx.w a0,@r5+
                            movx.w a0,@r5+
        rts
        nop

    ;Case 2: non-zero input vector
    VSCALE1_check_NLSeq31_end:
        add   #-14, r7      ;r7=MLS-14
        add   r7, r0        ;r0=NLS = NLSmin + (MLS-14)
        lds   r0, y0

    psha #16,y0
                            movs.w @r4+,x0
        psha x0,y0,a0       movx.w @r4+,x1
        psha x1,y0,a1       movx.w @r4+,x0
                            movs.w a0,@r5+
        psha  x0,y0,a0      movx.w a1,@r5+
                            movx.w a0,@r5+
                            movx.w @r4+,x1
        psha  x1,y0,a1      movx.w @r4+,x0
        psha  x0,y0,a0      movx.w a1,@r5+
                            movx.w a0,@r5+

        rts
        nop
```